AD-A031 802 BDM CORP ALBUQUERQUE N MEX F/G 9/2
SCEPTRE/LOGIC CIRCUIT ANALYSIS PROGRAM USER'S MANUAL.(U)
JUL 76 A F MALMBERG, L D RAY F29601-74-C-0106
UNCLASSIFIED BDM/A-145-75-TR-R1 AFWL-TR-75-279 NL

1 OF 2
AD A031802

END CONT.

DATE FILMED 12-76

AFWL-TR-75-279

AFWL-TR-
75-279

(2)

SCEPTRE/LOGIC CIRCUIT ANALYSIS PROGRAM USER'S MANUAL

BDM Corporation
2600 Yale Blvd., S.E.
Albuquerque, NM 87106

July 1976

Final Report

Approved for public release; distribution unlimited.

This research was sponsored by the Defense Nuclear Agency under Subtask Z99QAXTB027, Work Unit 51, Work Unit Title: Transient Radiation Response Models for Integrated Circuits.

Prepared for
Director
DEFENSE NUCLEAR AGENCY
Washington, DC 20305

AIR FORCE WEAPONS LABORATORY
Air Force Systems Command
Kirtland Air Force Base, NM 87117

6504

AFWL-TR-75-279

This final report was prepared by the BDM Corporation, Albuquerque, New Mexico, under Contract F29601-74-C-0106, Job Order WDNE1912 with the Air Force Weapons Laboratory, Kirtland Air Force Base, New Mexico. Robert G. Simon (ELP) was the Laboratory Project Officer-in-Charge.

When US Government drawings, specifications, or other data are used for any purpose other than a definitely related Government procurement operation, the Government thereby incurs no responsibility nor any obligation whatsoever, and the fact that the Government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation or conveying any rights or permission to manufacuture, use, or sell any patented invention that may in any way be related thereto.

This technical report has been reviewed and is approved for publication.

Robert G. Simon
ROBERT G. SIMON
Project Officer

FOR THE COMMANDER

Larry W. Wood
LARRY W. WOOD
Lt Colonel, USAF
Chief, Phenomenology/Technology Branch

Charles E. Ostrom
JAMES L. GRIGGS, JR.
Colonel, USAF
Chief, Electronics Division

Approved for public release; distribution unlimited.

DO NOT RETURN THIS COPY. RETAIN OR DESTROY.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: AFWL-TR-75-279	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): SCEPTRE/LOGIC CIRCUIT ANALYSIS PROGRAM USER'S MANUAL.	5. TYPE OF REPORT & PERIOD COVERED: Final Report.
6. PERFORMING ORG. REPORT NUMBER: BDM/A-145-75-TR-R1	
7. AUTHOR(s): Allan F. Malmberg, Larry D. Ray	8. CONTRACT OR GRANT NUMBER(s): F29601-74-C-0106
9. PERFORMING ORGANIZATION NAME AND ADDRESS: BDM Corporation, 2600 Yale Blvd., S.E., Albuquerque, NM 87106	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 62704H, WDNE1912
11. CONTROLLING OFFICE NAME AND ADDRESS: Director, Defense Nuclear Agency, Washington, DC 20305	12. REPORT DATE: Jul[illegible]76
	13. NUMBER OF PAGES: 52
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): Air Force Weapons Laboratory, Kirtland Air Force Base, NM 87117	15. SECURITY CLASS. (of this report): Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

This research was sponsored by the Defense Nuclear Agency under Subtask Z99QAXTB027, Work Unit 51, Work Unit Title: Transient Radiation Response Models for Integrated Circuits.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Computer Aided Analysis
SCEPTRE
Logic Simulation
Circuit Analysis

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

SCEPTRE/LOGIC is a modified version of the SCEPTRE circuit analysis program. This manual is intended as a guide for SCEPTRE users who wish to utilize the LOGIC features of the modified code. It is written with the assumption that the reader is familiar with input preparation and application of SCEPTRE (~~ref. 1~~).

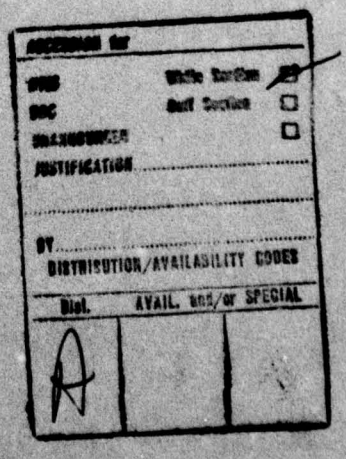
DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE When Data Entered

391 884 LB

TABLE OF CONTENTS



SECTION I
INTRODUCTION

SCEPTRE/LOGIC is a modified version of the SCEPTRE circuit analysis program. This manual is intended as a guide for SCEPTRE users who wish to utilize the LOGIC features of the modified code. It is written with the assumption that the reader is familiar with input preparation and application of SCEPTRE (ref. 1).

In the modified program, digital elements are recognized as part of the SCEPTRE input. The digital elements are grouped into defined units (modules) similar to defined models in SCEPTRE. A subprogram is automatically written to the subprogram tape for each module the user defines. During input processing, special recognition of calls to LOGIC subprograms is made so that association of initial conditions statements with the corresponding subprogram calls can be made.

---

[1] Sedore, S. R. and J. R. Sents. SCEPTRE Revised User's Manual, AFWL-TR-69-77, Vol. 1, July 1970.

SECTION II
THE LOGIC LANGUAGE

The LOGIC language has been constructed to conform closely with the SCEPTRE language structure. However, exceptions to this structure exist. These exceptions are stated explicitly as appropriate in the following discussion. The primary exception regards the construction of node names; all node names in LOGIC modules must be acceptable as FORTRAN variable names to the FORTRAN compiler in use. A sufficient set of conditions for this requirement is that the names begin with an alphabetic character, contain only alphabetic and numeric characters, and consist of six or fewer characters.

Rules regarding continuation of statements and multiple statements on a single card are the same as for SCEPTRE, except where modifications are stated explicitly.

As in SCEPTRE, card columns 1-72 are scanned for input information, leaving columns 73-80 free for comments or sequencing. All blanks are deleted prior to attempting syntax recognition. In figure 1, a portion of a SCEPTRE/LOGIC deck is included to illustrate the hierarchy of subheadings and to provide examples for the discussion below.

1. THE LOGIC DESCRIPTION SUBHEADING

A LOGIC DESCRIPTION subheading indicates that one or more logic module definitions follow. This subheading (and the associated module description cards) can appear in the CIRCUIT DESCRIPTION section or the MODEL DESCRIPTION section of a SCEPTRE deck. The hierarchical level of the LOGIC DESCRIPTION subheading is the same as that of the ELEMENTS subheading. The LOGIC DESCRIPTION subheading may be repeated within a given circuit description or model description. The order of appearance of the subheading relative to the other subheadings at the same level in the hierarchy is not important.

```
CIRCUIT DESCRIPTIØN

    LØGIC DESCRIPTIØN
    LØGIC FLØGIC(IN, ENABLE, THRESHØLD)
      LØGIC INPUTS
          N1 = IN .GT. THRESHØLD
          ENABLE = ENABLE .GT. 0.5
          NØT EN = ENABLE .LE. 0.5


      LØGIC ELEMENTS

          AND1, A1 - N1 - ENABLE
          AND2, A2 - NØTEN - ØUT
          ØR1, ØR1 - A1 - A2
          DELAY, ØUT - ØR1 = 1


      LØGIC ØUTPUTS

          ØUT = 0, 2.5
          ØUT = 2.5, 0
```

Figure 1. An Example of a LØGIC Module Definition

2. THE MODULE NAME CARD

The first card of each module definition assigns the name of the module (which will also be the subprogram name) and lists the names of the input interface dummy variables. The following card would initiate the definition of a module named FLOGIC, with three inputs (IN1, IN2, and IN3).

```
LOGIC FLOGIC (IN1, IN2, IN3)
```

The module name is used as the subprogram name; therefore, it must be an acceptable FORTRAN subprogram name, and must not duplicate existing SCEPTRE/LOGIC subprogram names or system subprogram names. All LOGIC internal variables begin with the sequence LL or the sequence YY. These sequences should not be used for the first letters of module names. Within the generated subprograms, references are made to EXIT, AMOD, IABS, AND and OR. These names cannot be used as module names.

The restrictions regarding dummy variable names are fewer, because they are not passed to the compiler. These names must begin with alphabetic characters, and must contain only alphabetic and numeric characters.

After the module name card, three subheadings (LOGIC INPUTS, LOGIC *OUTPUTS*, and *LOGIC ELEMENTS*) may appear. *The order of appearance is not* important, but each may appear no more than one time within the definition of a single module.

3. LOGIC INPUTS

The permitted values for the states of the nodes of a digital network are true and false. The interface from the continuum of values permitted for SCEPTRE variables to the true/false space of the digital network is defined by input interface statements under the LOGIC INPUTS subheading. The form of the input interface statement is

node name = a_1 op a_2

where a_1 and a_2 are numerical constants or dummy variable names, and op is a FORTRAN relational operator (.EQ., .NE., .GT., .LT., or .LE.).

In figure 1, three examples of input interface statements are shown.

```
N1 = IN .GT. THRESHOLD
ENABLE = ENABLE .GT. 0.5
NOT EN = ENABLE .LE. 0.5
```

These statements specify that the digital node N1 is true if and only if the value of the SCEPTRE variable corresponding (in the calling sequence) to dummy variable IN is greater than the value of the SCEPTRE variable corresponding to dummy variable THRESHOLD. Similarly, the node ENABLE is true when the SCEPTRE quantity corresponding to dummy variable ENABLE is greater than 0.5, and NOT EN is true when ENABLE is less than or equal to 0.5

There is no prohibition against the simultaneous use of a name as a node name, a dummy variable name, and an element name. A module name cannot be used in any other context within the same module, however. The order of appearance of the input interface statements is not important.

4. LOGIC OUTPUTS

After completion of the solution of the digital networks at a given simulation time, each digital node has a true or false value assigned. The output interface statement permits the user to designate the nodes for which values are returned to SCEPTRE; it also defines the numerical values corresponding to the logical states of these nodes.

The form of the output interface statement is

node name = number, number

The node name is the name of the node in the digital network for which it is desired to communicate results to the SCEPTRE network. The first number following the equality symbol is the value to be returned to SCEPTRE if the state of the digital node is false. The second number is the value to be returned when the state is true.

In the example in figure 1, two output interface statements are shown

```
OUT = 0, 2.5
OUT = 2.5, 0
```

When digital node OUT is false, a value of 0.0 will be returned to SCEPTRE for the value of the first output. A value of 2.5 will be returned when OUT is true. The second reference to subprogram FLOGIC will cause values of 2.5 or 0.0 to be returned for OUT values of false or true, respectively.

The order of appearance of the output interface cards must correspond to the order in which calls will be made from SCEPTRE. The number of calls from SCEPTRE must not exceed the number of output interface definitions. Further exposition on these requirements will be given in the section on calling LOGIC modules from SCEPTRE.

5. LOGIC ELEMENTS

The elements of a digital network are specified by a statement of the general form

element name, output - $input_1$ - $input_2$ - ... - $input_n$ = value

The element name is used to identify the element type (by inspection of the first letter of the name) and for assignment of initial conditions. (See the discussion on Initial Conditions below.) If an element is not referenced by an initial conditions statement, it is not required that the name of the element be unique from the names of the

other elements of the same type. The key letters and their corresponding elements are listed in figure 2.

The nodes of an element are listed after the comma which follows the element name. The first node in the list is always the output node. The number of input nodes which is permitted depends on the element type. For some elements the node list is followed by an equal sign and a value specification or an expression.

The node names are required to be acceptable FORTRAN variable names for the compiler in use. A sufficient set of conditions is that they contain only alphabetic and numeric characters, that the first character be alphabetic, and that the number of characters is no greater than six. The module name cannot be used as a node name, but dummy variable names and element names can be used as node names. Names beginning with LL and YY should be avoided to prevent conflicts with internal variables in the generated subprograms. The standard SCEPTRE control variable names, all of which (except TIME) begin with the letter X and contain three or more alphabetic characters, must also be avoided as node names. The FORTRAN functions EXIT, AMOD, IABS, AND, and OR are used in the generated subprogram, and these names are therefore not available as node names.

a. AND Element

The logical AND element has one output node and as many input nodes as desired. The element name always begins with the one-character prefix A which identifies the element type. The node list begins with the output node, followed by the input nodes.

Example:

```
AND1, A-VC1-GF
```

b. OR Element

The logical OR element is identified by the letter O as the first character. The node list begins with the output node followed by any number of input nodes.

Example:

```
OR1, B-AN12
```

Key Letter	Element
A	AND
O	OR
M	NAND
N	NOR
I	INVERTER
E	EXCLUSIVE OR
X	BOOLEAN EXPRESSION
Z	BOOLEAN EXPRESSION WITH FEEDBACK
B	FLIP-FLOP
D	DELAY
U	EDGE DETECTOR
T	LOGIC TRANSISTOR

Figure 2. Key Letters and Corresponding Logical Elements

3

c. NAND Element

The letter M as the first character of an element identifies that element as a NAND gate. The output node is listed first, followed by as many input nodes as desired.

Example:

M456, A-B-C-D

d. NOR Element

The NOR gate element is identified by an N as the first character. The output node is listed first, followed by the input nodes.

Example:

N456, A-B-C-D

e. Inverter Element

The inverter element accompishes the logical NOT function, has only one input, and is denoted by the prefix I.

Example:

I456, A-B

f. Exclusive OR Element

The exclusive OR gate is identified by an inital E and has two inputs.

Example:

E456, A-B-C

g. Boolean Function (Without Feedback)

An X as the first character of an element name means that the element is defined by a Boolean expression, and is not involved in a logical feedback loop.

Example:

X, X1-D-B = B.AND.D .OR. .NOT.B .AND. .NOT.D

The expression which follows the equal sign may be any logical-valued expression acceptable to the relevant compiler. No error checking or interpretation of the expression is attempted by the LOGIC input processor, except to ensure that left and right parentheses are paired. Continuation of the expression to the next card is implied by an unsatisfied operational symbol.

No recognition is made of (or prohibition made against) the use of an expression which includes the output node as an operand. If this is done, the value of the expression will be calculated using the state of the output node from the previous computation. The previous state may be from the previous time step, or it may be from the previous iteration, depending on the topology of the network. Note that the X element can be comported as a logical oscillator.

h. Boolean Function (With Feedback)

A Z as the first character of an element name means that the element is defined by a Boolean expression which includes the output of the element as an operand.

Example:

Z, Z1-D-B = Z1 .AND. D .OR. B

There is no time delay associated with the Z element. The computation of the output value is repeated until a self-consistent solution is obtained using the output value from the previous iteration, or until the iteration limit is exceeded. If the iteration limit is exceeded, a message is printed, and the simulation is aborted. Note that the Z element is not a sequential element in the classical sense even though its output is fed back.

The expression which follows the equal sign may be any logical-valued expression acceptable to the relevant compiler. No error checking or interpretation of the expression is attempted by the LOGIC input processor, except to ensure that left and right parentheses are paired. Continuation of the expression to the next card is implied by an unsatisfied operational symbol.

i. J-K Flip-Flop Element

The clocked J-K flip-flop has $\overline{\text{SET}}$ and $\overline{\text{RESET}}$ inputs and is denoted by B as the first character of the element. The nodes are written in the following order: output, J input, K input, $\overline{\text{SET}}$ input, $\overline{\text{RESET}}$ input, and CLOCK input. The initial state of the output node is specified under the normal INITIAL CONDITIONS subheading. The Karnaugh map and Boolean expression for this element are given in figure 3.

		C	0	0	0	0	1	1	1	1
		SB	0	0	1	1	0	0	1	1
Q_n	J	RB / K	0	1	0	1	0	1	0	1
0	0	0	0	1	0	0	0	1	0	0
0	0	1	0	1	0	0	0	1	0	0
0	1	0	0	1	0	0	0	1	0	1
0	1	1	0	1	0	0	0	1	0	1
1	0	0	0	1	0	1	0	1	0	1
1	0	1	0	1	0	1	0	1	0	0
1	1	0	0	1	0	1	0	1	0	1
1	1	1	0	1	0	1	0	1	0	0

SB = $\overline{\text{SET}}$

RB = $\overline{\text{RESET}}$

C = CLOCK

J = J INPUT

K = K INPUT

Q_n = Last computed value of Q

Q_{n+1} Karnaugh Map for Clocked J-K FLIP-FLOP with $\overline{\text{SET}}$ and $\overline{\text{RESET}}$ Inputs

$$Q_{n+1} = RB \cdot \overline{SB} + (SB \cdot RB) \cdot ((C \cdot J \cdot \overline{Q}_n) + (Q_n \cdot \overline{K \cdot C}))$$

$\cdot$ = .AND. $+$ = .OR.

Figure 3. Boolean Expression for J-K FLIP-FLOP as Implemented in LOGIC

Example:

B45, A-VJ-VK-VSB-VRB-VC

j. Delay Element

D denotes the delay element. The first node is the output. The second node is the input. There may be only one input. A delay time may be specified following the node specification. If a value is specified, the element is a time delay element; otherwise, it is a computational delay element.

Examples:

D456, OUT - IN (Computational Delay)

D456, OUT - IN = .01 (Time Delay)

A computational delay is always satisfied within the current time step. If a transition occurs at the input of a computational delay element, an entry is made into a special first-in-first-out queue. As this queue is emptied, the output of the appropriate delay is complemented and the state of each element affected by the transition is recomputed. At time zero, and during initial conditions calculations, the calculation is repeated until a consistent solution is obtained with the input of each delay equal to the output.

For a time delay, separate value specifications may be given for the rise and fall transitions. When two values are given, the first value is the risetime and the second is the fall time. The value pair must be enclosed in parentheses and separated by a comma.

Example:

D456, OUT - IN = (.1, .2)

Delay value specifications may be given as arithmetic expressions. An expression may be any numerical-valued expression acceptable to the relevant compiler. No error checking or interpretation of expressions is attempted by the LOGIC input processor, except to ensure that left and right parentheses are paired. Continuation of an expression to the next card is implied if a numerical operator is the last symbol on a card.

Examples:

```
D1, D1 - IN = XSTPSZ*0.1
D2, D2 - IN = (XSTPSZ*0.1, XSTPSZ*0.1 + 1.)
D3, D3 - A = 1.0 + FLOAT(SHIFT(AND(NODE1,
             4000000000000000000B), 1))
D, D - A = (YYRG3, YYRG4)
```

1) Delay Buffer Length

When the input of a time delay element changes state, an entry is made into a sorted list of timed events. The lowest valued time in the list is the next time at which the LOGIC network must be calculated (in addition to the SCEPTRE integration times). The length of the delay buffer must be sufficient to accommodate all time delays which may be active at the same time, including any multiple transitions. This buffer also includes a list of active computational delays and edge detectors. The computational delay list is emptied on a first-in-first-out basis. The default value for the length of the delay buffer is sufficient to store a number of transitions equal to twice the total number of delays and edge detectors from all LOGIC modules. If the buffer becomes full during the simulation, the following message will be printed.

```
INSUFFICIENT DELAY BUFFER SPACE - RUN TERMINATED -
        CURRENT BUFFER LENGTH IS XXXXXX
```

The length of the buffer may be changed by placing a buffer length specification card after a LOGIC DESCRIPTION card and before the first LOGIC name card.

```
LOGIC DESCRIPTION
BUFFER LENGTH = number
    LOGIC LOG1 (A, B, C)
      .
      .
      .
    LOGIC LOG2 (A, B, C)
      .
      .
      .
```

The number specified replaces the default value. The default value is twice the sum of the number of delays and edge detectors in all modules. Modularity may be maintained by repeating the LOGIC DESCRIPTION card and buffer length card for each module.

```
LOGIC DESCRIPTION
BUFFER LENGTH = number_1
     LOGIC LOG1 (A, B, C)
          .
          .
          .

LOGIC DESCRIPTION
BUFFER LENGTH = number_2
     LOGIC LOG2 (A, B, C)
          .
          .
          .
```

In this case, the length of the buffer will be the sum of the lengths specified on the buffer length cards.

2) Sequencing of Time-Dependent Events

At time zero, the entire network which represents a given module is solved for a self-consistent solution with the output of each delay equal to the input. After time zero, the state of the combinatorial network originating at the inputs is calculated at each SCEPTRE time step. If the input of a time delay changes state, an entry for that delay is made into the event queue. When SCEPTRE completes a time step at time t, the event queue is checked for events at time less than or equal to time t. For each such event, reentry to the appropriate LOGIC subprogram is made, and the state of the combinatorial network originating at the output node of the delay is calculated.

This method has the advantage that the state of an inactive subnetwork of a sequential network is not recalculated on each SCEPTRE time step, but is only calculated when a transition occurs to change the state of that subnetwork. This results in a considerable increase in efficiency for some networks. Since sources can occur only

as SCEPTRE electrical elements, any transitions within a logical network must have originated at the input interfaces driven by SCEPTRE. This is the justification for the "SCEPTRE-first" philosophy.

If time delays are used only for internal sequencing within a logical network, the SCEPTRE time step can be allowed to overstep the logical time delays. That is, the SCEPTRE time and LOGIC time may be allowed to progress nonsynchronously. However, if time-dependent events external to a logic module must be synchronous with time delayed events inside the module, the user must specify a maximum step size for SCEPTRE which is consistent with the degree of synchronization error he wishes to permit.

k. Edge Detector - Differentiator

The U, used as the first character of an element name, identifies the element as an edge detector or differentiator. A leading edge detector is signified by a T or 1 in the value field and a trailing detector is signified by an F or 0 in the value field.

Example:

U456, A-B = T

The "pulse width" of the edge detector is zero. That is, the down transition of the output is a second iteration within the same time step as the up transition. If a nonzero pulse width is desired, an edge detector can be modeled with a time delay and a Boolean element.

D, D - IN = 1

X, U - IN - D = IN. AND. .NOT. D

The edge detector causes entries to be stored in the computational delay list when an edge is detected. Therefore the delay buffer (section 5.j.(1)) must be long enough to accommodate the edge detectors. Normally, the default size of this list is sufficient, and the user need not be concerned with the length of the list.

l. LOGIC Transistor Element

Four transistor configurations may be modeled: transistor amplifier, common emitter stage, emitter follower, and an emitter input

transistor stage. An element name for a transistor begins with the letter T. The node list begins with the output node(s) followed by the input nodes as with other elements; however, for the transistor element, prefixes are used to indicate the function of each node. The collector node name is prefixed by the letter C, the emitter by the letter E, and the base by the letter B. To further differentiate output and input nodes, the input node list always begins with the base as the first input node. The single letter prefixes are used for node function identification only; they are not part of the node name.

The radiation input, if used, may either turn an "off" transistor "on" or turn an "on" transistor "off." The desired effect is selected by prefixing the radiation input with the letter F or T (used only in the logic transistor element) where F indicates an "on" transistor will be turned "off" by the radiation.

The four configurations are discussed below.

1) Transistor Amplifier

The transistor amplifier has two outputs, one taken from the collector and one from the emitter; an input to the base; and an optional radiation input. The collector output must appear first in the node list.

Example:

T234, CM5-EN6-BP7-FRAD

An "on" transistor has the collector in the logical false state and the emitter in the true state. An "off" transistor has the logical states of the two outputs reversed. In the absence of a radiation node specification or when the state of the radiation input is false, the transistor is always "on" when the base input is true, and "off" when the base input is false.

When the radiation prefix is F, the Boolean equivalent of the transistor amplifier is

$$C = \overline{B} \cdot \overline{R}$$

$$E = \overline{C}$$

When the radiation prefix is T, the Boolean equivalent is

$$C = \overline{B} + R$$

$$E = \overline{C}$$

An example of a transistor amplifier stage is given in figure 4.

2) Common Emitter Stage

The common emitter stage has one output taken from the collector, one input to the base, and an optional radiation input.

Example:

T345, CM5-BP7-TRAD

A common emitter stage is shown in figure 5. For a logic 0 input to base, the collector is near supply potential, which gives a collector output of logic 1. For a logic 1 input to the base, the transistor turns on, pulling the collector to a logic 0.

When the radiation prefix is F, the Boolean function is

$$C = \overline{B} \cdot \overline{R}$$

When the prefix is T, the Boolean function is

$$C = \overline{B} + R$$

3) Emitter Follower

The emitter follower has an emitter output, a base input, and an optional radiation input.

Example:

T456, EN6-BP7-FRAD

An emitter follower stage is shown in figure 6. For a logic 0 input to the base the transistor is "off," and the emitter logic output is 0. For a logic 1 input to the base the transistor is "on," and the emitter logic output is 1.

When the radiation prefix is F, the equivalent Boolean function is

$$E = B + R$$

When the prefix is T, the function is

$$E = B \cdot \overline{R}$$

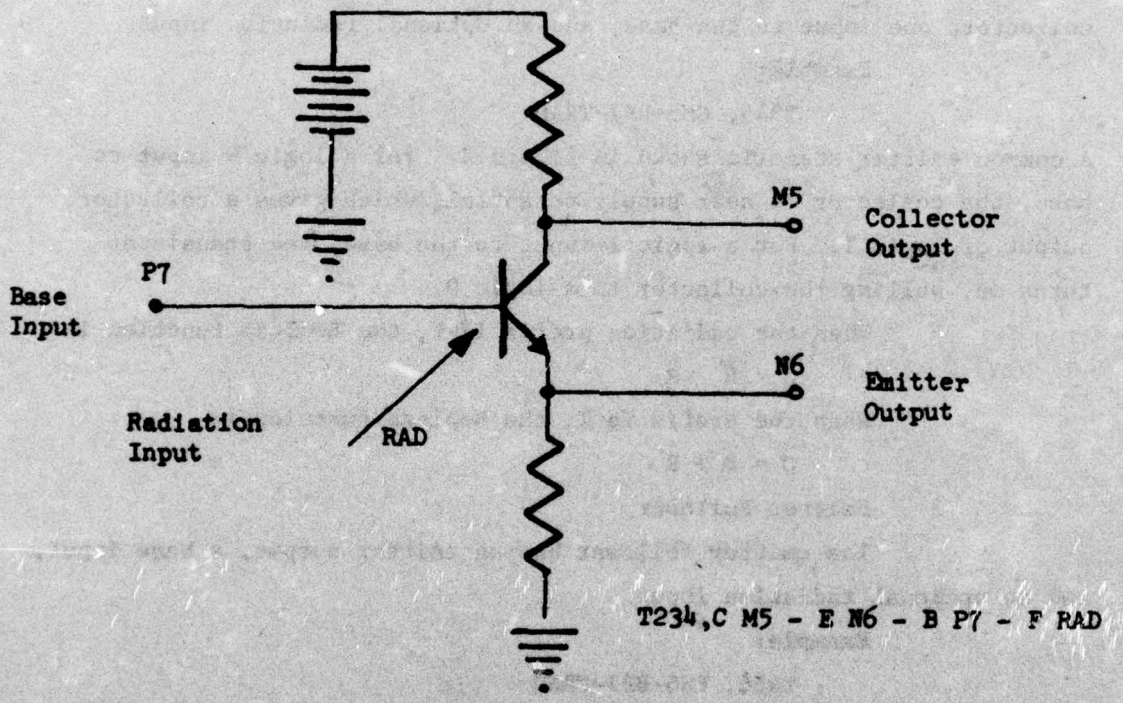


Figure 4. A Transistor Amplifier

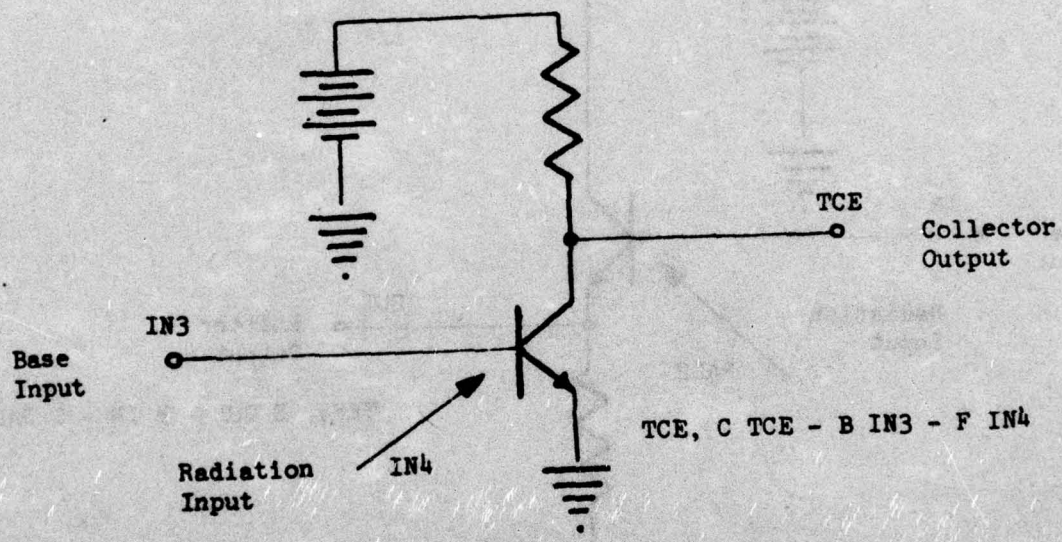


Figure 5. A Common Emitter Stage

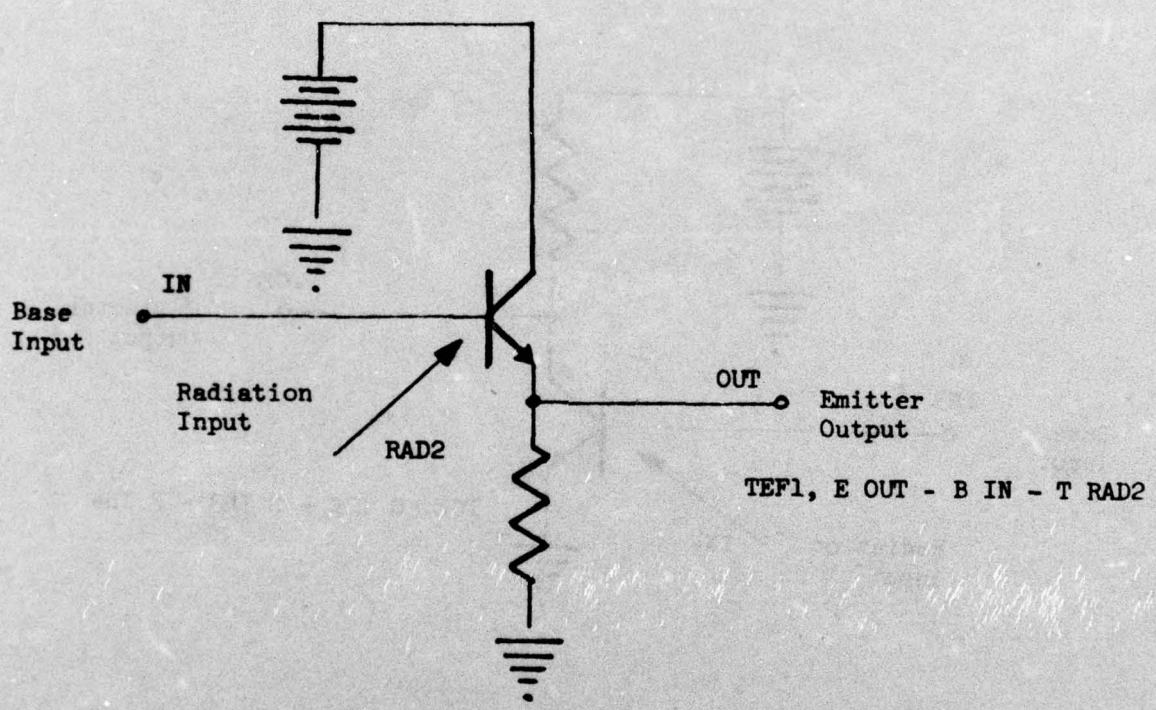


Figure 6. An Emitter Follower Stage

4) Emitter Input Transistor Stage

The emitter input transitor stage has a collector output, base and emitter inputs, and an optional radiation input.

Example:

T567, CM5-BP7-EN6-TRAD

For a logic 0 input to the emitter, a logic 0 on the base will force the collector output to be a logic 1. This is the transistor "off" state. With the emitter at logic 0, a logic 1 on the base turns the transistor "on" and the collector output goes to logic 0. If the emitter input is a logic 1 the collector output remains a logic 1 independent of base or radiation state. The radiation input will affect the output only when the emitter input is a logic 0. Then the radiation logic 1 input will turn the "off" transistor "on" for a prefix F and the "on" transistor "off" for a T prefix.

When the radiation input prefix is F, the equivalent Boolean expression is

$$C = \overline{B} \cdot \overline{R} + E$$

When the prefix is T, the Boolean expression is

$$C = \overline{B} + R + E$$

An emitter input stage is shown in figure 7.

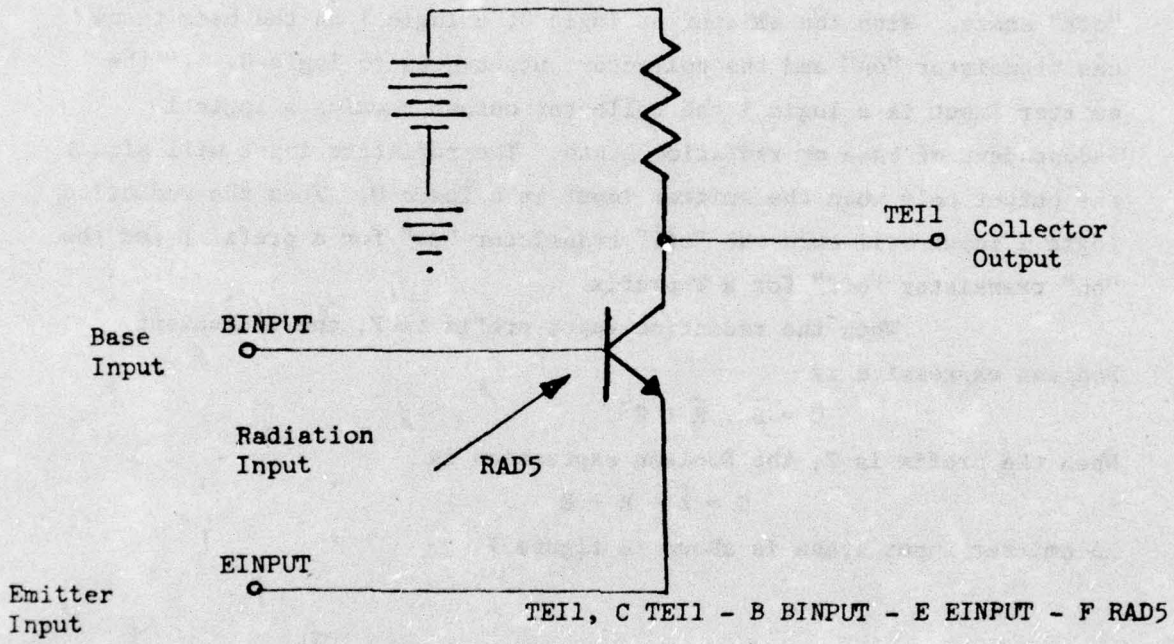


Figure 7. A Transistor Stage With Emitter Input

SECTION III
CALLING LOGIC SUBPROGRAMS FROM SCEPTRE

During processing of the SCEPTRE/LOGIC input deck a copy of each unique LOGIC module definition is saved on a file. When processing of the remainder of the CIRCUIT DESCRIPTION information is completed, the LOGIC input for each module is processed, and a subprogram is written to the SCEPTRE subprogram tape for each module. The references to these subprograms by the SCEPTRE circuit are made as with other subprograms, with one important exception.

The first argument in the call to a LOGIC subprogram is always an integer constant. This constant, the module replication index, is used by the subprogram to identify the individual calls in a sequence of repetitious calls of a single defined module. Frequently, the original order of the calls is altered by SCEPTRE. The module replication index is also used to associate initial conditions with the appropriate repetition of a module call. (See the discussion on initial conditions below.)

Figure 8 shows a partial SCEPTRE/LOGIC input deck which illustrates the method of calling LOGIC subprograms. In the examples shown here, modules are called by means of defined equations, because this is the most general method. Note that the first dummy variable in the equation variable string used to call the LOGIC subprogram must always be a letter variable. For example, A is used in the first position in equation Q LOG in figure 8. A module may also be called as an external function if the module name begins with the letter F. However, LOGIC modules should not be called from expressions.

```
CIRCUIT DESCRIPTIØN
     LØGIC DESCRIPTIØN
     LØGIC FLØGIC(IN, ENABLE, THRESHØLD)
          LØGIC INPUTS
             N1 = IN .GT. THRESHØLD
             ENABLE = ENABLE .GT. 0.5
             NØT EN = ENABLE .LE. 0.5
          LØGIC ELEMENTS
             AND1, A1 - N1 - ENABLE
             AND2, A2 - NØTEN - ØUT
             ØR1, ØR1 - A1 - A2
             DELAY, ØUT - ØR1 = 1
          LØGIC ØUTPUTS
             ØUT = 2.5, 0
     ELEMENTS
         E1, GND - E1 = QLØG (1, VC1, P1, 1.0)
              .
              .
              .
         E2, GND - E2 = QLØG (2, VC21, P2, 1.0)
              .
              .
              .
   FUNCTIØNS
      QLØG(A,B,C,D) = (FLØGIC(A,B,C,D))
              .
              .
              .
```

Figure 8. An Example of Multiple Calls to a LØGIC Subprogram

1. THE MODULE REPLICATION INDEX

The first argument in the argument list of each call

```
E1, GND - E1 = QLOG(1, VC1, P1, 1.0)
                .
                .
                .
E2, GND - E2 = QLOG(2, VC21, P2, 1.0)
```

is the module replication index. The remainder of the argument list is in one-to-one correspondence with the dummy variable list on the module name card.

```
LOGIC FLOGIC (IN, ENABLE, THRESHOLD)
```

The module replication index is used to identify each repetition of a LOGIC module in a manner analogous to the usage of the element name associated with each cell to a SCEPTRE defined model.

```
T1, A-B-C-D = MODEL XYZ
T2, E-FG-H = MODEL XYZ
              .
              .
              .
```

2. MODULES WITH MULTIPLE OUTPUTS

If a LOGIC module has more than one output (more than one output interface card under the LOGIC OUTPUTS subheading), the calling sequence is as illustrated in figure 8. The first call for each occurrence of the module is made through an equation, and the actual input parameters are passed in the argument list. The values for all defined outputs are calculated on the first entry (at each integration pass) with each new value of the call index. The output values are saved in an internal array. On subsequent entries to the subprogram, if the index value has already occurred during the same integration pass, the appropriate value in the sequence of outputs is delivered. The order in which SCEPTRE

generates the cells is not necessarily the same as the order in which the cards appear in the input deck. Therefore it is necessary to introduce dependencies to force the correct order of the calls. In the example of figure 9, J1 is placed in the argument list for the call to obtain the value of J2, and J2 is placed in the argument list for the calculation of the value of J3. This forced ordering ensures that J1 will have the value corresponding to the first output interface statement, J2 will have the value corresponding to the second, and J3 will have the value corresponding to the third.

There is no need to introduce dependencies between J1 and J21, or between J1 and J31, because the order (relative to each other) in which SCEPTRE writes these calls is irrelevant. In fact, the calls corresponding to the sets (J1, J2, J3), (J21, J22, J23), and (J31, J32, J33) may be intermixed as long as the order within each set is maintained.

3. COMPUTATIONAL DELAYS

A note of caution is appropriate at this point relative to computational delays. If the SCEPTRE dependency analysis reveals the necessity of introducing a computational delay (not to be confused with the computational delay element in LOGIC), the order of subprogram calls chosen by SCEPTRE may not correspond to the order indicated by the element dependencies. This may result in erroneous results from the LOGIC subprogram. Computational delays will not occur if equation and function arguments (including input variables to LOGIC calls) are limited to capacitor voltages and inductor currents. The validity of use of other variables is topology dependent. For additional information on this subject, the user is referred to the SCEPTRE manual.

```
CIRCUIT DESCRIPTIØN

   LØGIC DESCRIPTIØN
   LØGIC LØG2(A, B, C)
      LØGIC INPUTS
         .
         .
         .
      LØGIC ELEMENTS
         .
         .
         .
      LØGIC ØUTPUTS
        X = 0, 1
        Y = 0, 1
        Z = 0, 1

FUNCTIØNS

      Q1(NDEX, A, B, C) = (LØG2(NDEX,A,B,C))
      Q2(NDEX, DEP) = (LØG2(NDEX, DEP, 0., 0.))

ELEMENTS

      J1, J1 - G = Q1(1, VC1, VC2, VC3)
      J2, J2 - G = Q2(1, J1)
      J3, J3 - G = Q2(1, J2)

      J21, J21 - G = Q1(2, VC21, VC22, VC23)
      J22, J22 - G = Q2(2, J21)
      J23, J23 - G = Q2(2, J22)

      J31, J31 - G = Q1(3, VC31, VC32, VC33)
      J32, J32 - G = Q2(3, J31)
      J33, J33 - G = Q2(3, J32)
         .
         .
         .
```

Figure 9. Example of Multiple Calls to a LØGIC Module with Multiple Outputs

4. REFERENCING LOGIC MODULES DEFINED IN ANOTHER MODEL

If a LOGIC module is to be referenced by two or more SCEPTRE models (or by the main circuit and a model), it is not necessary to duplicate the module definition. However, it is necessary to identify the name as a LOGIC module name by a special declaration

LOGIC name (EXT)

This card appears in place of the regular LOGIC name card. It is not followed by a dummy argument list or any other subheadings.

In the example in figure 10, LOGIC module ABC is defined in model AND2 and called from AND2 and from the main circuit.

In this example, two different calls to subprogram ABC are generated. The first is from equation QABCT1, and the second is from equation QA9 in the main circuit. The single-card entry under the LOGIC DESCRIPTION subheading in the main circuit description identifies ABC as the name of a module which is defined external to the current level of definition.

The module replication index values must be unique within the main circuit description and within each model. However, the module sequencing can begin at unity within each model.

```
MØDEL DESCRIPTIØN
MØDEL AND2(TEMP) (A-B-C-D)
   ELEMENTS
      E1, A - B = QABC(1, VC1, VC2)
       .
       .
       .
   FUNCTIØNS
      QABC(N, A, B) = (ABC(N, A, B))
       .
       .
       .
   LØGIC DESCRIPTIØN
      LØGIC ABC(A,B)
         LØGIC INPUTS
              A = A.GT.1
              B = B.GT.1
         LØGIC ELEMENTS
              AND, C - A - B
              DELAY, D - C = .1
         LØGIC ØUTPUTS
              D = 0, 10

CIRCUIT DESCRIPTIØN

   ELEMENTS
      E1, E1 - G = QA(1, VCX, VCY)
      T1, C1 - C2 - X1 - X2 = MØDEL AND2(TEMP)
       .
       .
       .
   FUNCTIØNS
      QA(N, A, B) = (ABC(N, A, B))
   LØGIC DESCRIPTIØN
      LØGIC ABC(EXT)
   RUN CØNTRØLS
     .
     .
     .
```

Figure 10. Illustration of a Reference to a Module Defined in a Different LØGIC DESCRIPTIØN

SECTION IV
INITIAL CONDITIONS

Initial conditions statements for LOGIC elements are included under the usual SCEPTRE subheading, INITIAL CONDITIONS. LOGIC initial conditions statements may be intermixed with SCEPTRE initial conditions statements. Multiple statements, separated by commas, may be written on a single card. An initial conditions statement cannot be continued to the next card, however.

Initial conditions may be specified only for those LOGIC elements for which the current value of the output depends on a past value. These elements are the delay, flipflop, edge detector, and the Boolean element with feedback. Therefore if the initial letter of an initial conditions entry is not V or I (for a regular SCEPTRE entry), and is not D, B, U, or Z, the program will deliver an error message.

The effect of an initial conditions specification for a given logical element is to preset the state of the output node of that element to the specified value.

If an initial conditions specification is made from within a model definition, or if it is made within the main circuit description and does not reference an element outside the main circuit description, the format is

Element name (Module replication index, Module name) = Value

The digit 0 or the letter F may be used to specify a false value. The digit 1 or the letter T may be used to specify a true value.

```
DELAY(1, ABC) = T
DELAY(2, ABC) = F
B1(1, XYZ) = 1
```

The default state for all nodes is false. However, permitting a node to have a default false state is not equivalent to setting a false state by an initial condition. The default state will be changed automatically if it is inconsistent with other initial states. A specified initial condition cannot be changed automatically and will cause an aborted run if it is inconsistent.

If only one module name is known to the model (or main circuit) from which the initial conditions entry is made, the module name may be omitted.

```
DELAY(1) = T
DELAY(2) = F
```

If the highest value of the replication index (within the current description level) is unity, and if the module name can be omitted (as described above), then the index may also be omitted.

```
DELAY = T
```

The initial states for elements in models can be specified by initial conditions specifications in the main circuit description. It is necessary, in this case, to specify the element prefix by which the model was called into the main circuit. The format is

```
Element name (Module replication index, Module name,
                Element prefix) = Value
```

For example, in figure 10, the model AND2 is called into the main circuit with the element prefix, T1. Model AND2 includes a reference to LOGIC module ABC. The following statements could be added under the CIRCUIT DESCRIPTION heading.

```
INITIAL CONDITIONS
  DELAY(1, ABC) = T
  DELAY(1, ABC, T1) = T
```

The first initial conditions specification refers to the module call which is made from the main circuit. The second refers to the call from within model AND2. If an initial condition is set on a module called into the circuit multiple times, the initial condition should be specified for all calls to the module.

Further examples of cross-referencing of LOGIC modules and of initial conditions specifications are shown in figure 11.

Two notes of caution are appropriate regarding the specification of initial conditions. If a specified nodal state is inconsistent with the calculated state based on the circuit description and the initial states of the other nodes in the circuit, the run will be aborted with the message "cannot obtain selfconsistent DC solution from module (name) run aborted." Secondly, if a logically impossible loop is described, LOGIC will deliver a message stating that convergence could not be achieved in a loop. At time zero, LOGIC iterates until a solution is obtained with the output of every delay equal to the input of that delay. Therefore, a circuit intended to oscillate during the transient analysis may not achieve convergence at time zero.

```
MØDEL DESCRIPTIØN
MØDEL AND3(PERM) (A-B-C-D-E-F-G)

ELEMENTS

E1,G-D = QAND(1, VR1, VR2, VR3)
E2,G-12 = QAN(1, E1)

E10,G-F = QAND (2, VR10, VR20, VR30)
E20,G-120 = QAN(2, E10)
  .
  .
  .
LØGIC DESCRIPTIØN
LØGIC AND3(ARG1, ARG2, ARG3)

LØGIC INPUTS
A = ARG1.LT.-5.
B = ARG2.LT.-5
C = ARG3.LT.-5.

LØGIC ELEMENTS
AND, D-A-B-C
DELAY, E-D = (.1,.2)

LØGIC ØUTPUTS
E = 0, 10
E = 10, 0

FUNCTIØNS
QAND (A,B,C,D) = (AND3(A,B,C,D))
QAN(A,B) = (AND3(A,B,0,0))
  .
  .
  .
```


Figure 11. Example of Initial Conditions Specifications

```
MØDEL DESCRIPTIØN
MØDEL UNAMAS(TEMP) (A1,A2,A3,A4)

LØGIC DESCRIPTIØN
LØGIC AND2(EXT)

ELEMENTS
E1,A-B=Q1(1,VR1,VR2)
 .
 .
 .
FUNCTIØNS
Q1(A,B,C) = (AND2(A,B,C))
 .
 .
 .

CIRCUIT DESCRIPTIØN

LØGIC DESCRIPTIØN

LØGIC AND3 (EXT)

LØGIC AND2(ARG1, ARG2)
LØGIC INPUTS
A=ARG1.GT.0.5
B=ARG2.GT.0.5
LØGIC ELEMENTS
AND,C-A-B
DELAY,D-C=(0.1,0.2)
LØGIC ØUTPUTS
D=0,10

ELEMENTS
E1,G-1=QAND2(1,VR1,VR2)
E2,G-2=QAND2(2,VR21,VR22)
A1,A-B-C-D-G-E-F-... = MØDEL AND3(PERM)
A2,AA-BB-CC-DD-GG-EE-FF-... = MØDEL AND3(PERM)

E53,53-G=QAND3(1,VR53,VR54,VR55)
E64,64-G = QAN(1,E53)

A3,N1-N2-N3-N4=MØDEL UNAMAS(TEMP)
 .
 .
 .
```


Figure 11. Example of Initial Conditions Specifications (Continued)

```
INITIAL CØNDITIØNS
VC1=0
DELAY(1,AND2)=T, DELAY(2,AND2)=T
DELAY(1,AND3,A1)=T, DELAY(1,AND3,A2)=T, DELAY(2,AND3,A1)=T
DELAY(1,AND2,A3)=T
DELAY(1,AND3)=T

FUNCTIØNS
QAND2(A,B,C)=(AND2(A,B,C))
QAND3(A,B,C,D)=(AND3(A,B,C,D))
QAN(A,B)=AND3(A,B,0,0))
  .
  .
  .
```


Figure 11. Example of Initial Conditions Specifications (Concluded)

SECTION V
PROBLEM SIZE LIMITATIONS

The array sizes in the program, as distributed, limit the user to 50 separate module definitions, or 100 calls to a single module, or an intermediate combination. The number of LOGIC initial conditions statements is limited to 50.

Approximately 500 elements can be accomodated in a single module without changing internal array sizes. The number of elements depends on element type, because some elements require more storage than others.

The sum of time delays and Boolean elements is limited to 500 or less, depending on the length of the value expressions. Double-valued delay elements occupy twice the table space of single-valued delay elements; therefore a maximum of 250 double-valued delays is permitted.

SECTION VI
THE TIME EVENT QUEUE

As described in Section V, delay information is stored only when the delay is active; that is, when a transition has occurred at the input of a delay, and the delay time has not expired. For each such transition, an entry is stored in the time event queue.

The technique utilized for this queue involves storing information in a quasi-sorted order in a bifurcated arborescence which is dubbed a leftist tree (so called because it leans to the left, i.e., there are predominantly more links to the left than to the right). The algorithm for manipulating such a tree is such that the top cell of the tree (the root) is always guaranteed to be the cell with the earliest future time, even though all other cells are only in quasi-sorted order.

Each cell contains the transition output time, the propagation delay identity, a path distance value, a pointer to a left subtree, and a pointer to a right subtree. This information is stored using two words (by means of packing) so that the space requirements are 2n where n is the number of delays active.

Since the transition output time is the quantity upon which sorting is based, it will be referred to as the KEY in the following discussion. The path distance D is the minimum path length from the node to a leaf of the tree; because of the way in which the tree is constructed, this minimum length path will always be a right-most path. The pointer to the left subtree will be denoted by LP, and the pointer to the right subtree will be denoted by RP. A pointer to a leaf in the tree will be assigned a value of 0. The leaf does not actually contain any explicit information and therefore leaves are not actually represented in memory. We may refer to the KEY and D quantities for the cells comprising the roots of the left and right subtrees by KEY(LP), D(LP), KEY(RP), and D(RP). We adopt the convention that KEY(0) = ∞ and D(0) = 0.

The leftist tree may be defined by listing the properties of the KEY and D fields for each cell P:

a. KEY(P) $\leq$ KEY(LP(P))

b. KEY(P) $\leq$ KEY(RP(P))

c. D(P) = 1 + Min(D(LP(P)),D(RP(P)))

d. D(LP(P)) $\geq$ D(RP(P))

Removal of the root of a leftist tree (i.e., access to the earliest future time) requires a constant time; however, it must be followed by a merging of the two subtrees below the root before any other operations are executed on the tree. The subtree merging is not required if the right subtree is vacuous, and is trivial if the left subtree is vacuous. In all other cases, subtree merging requires 2(log n) time in the worst case. Insertion of a new cell into the tree also requires 2(log n) time in the worst case. In the best case, merging and insertion require a constant time. Furthermore, the software for merging and insertion is identical so that only a single routine is required for all tree manipulations. The overhead for small n is reasonable and the method is very efficient for large n.

The algorithm for merging two leftist trees is shown in Figure 12 and an example is given in Figure 13.

1. LET P AND Q BE THE TWO TREES, WHERE P HAS THE SMALLEST KEY.

2. LET X REPRESENT TREE P.

3. TRAVERSE TREE X ALONG THE RIGHT-MOST LINKS, COMPARING THE KEY AT EACH NODE OF X WITH THE KEY OF Q, SO AS TO LOCATE THE INSERTION POINT FOR Q ACCORDING TO ASCENDING KEY VALUE. SAVE NODES TRAVERSED ON STACK.

4. AT THE INSERTION POINT BREAK TREE X, SAVING THE SUBTREE OF X BELOW THE INSERTION POINT AS TREE T.

5. APPEND TREE Q TO TREE X AT THE INSERTION POINT, USING THE LEFT LINK IF IT IS NOT ALREADY IN USE; OTHERWISE USE THE RIGHT LINK.

6. IF TREE T IS EMPTY GO TO 8.

7. LET X REPRESENT TREE Q. LET Q REPRESENT TREE T. GO TO 3.

8. TRAVERSE THE MERGED TREE IN REVERSE ORDER, STARTING AT THE LAST INSERTION POINT AND ENDING AT THE ROOT, USING THE NODE SEQUENCE SAVED ON THE STACK TO RETRACE THE PATH. NOTE THE MINIMUM DISTANCE FROM EACH NODE TO A LEAF. IF THE RIGHT-MOST PATH IS LONGER THAN THE MINIMUM DISTANCE PATH TO A LEAF AT A GIVEN NODE, INTERCHANGE THE LEFT AND RIGHT SUBTREES AT THAT NODE. THE ALGORITHM TERMINATES AFTER THE ROOT NODE HAS BEEN PROCESSED.

Figure 12. Algorithm for Merging Two Leftist Trees

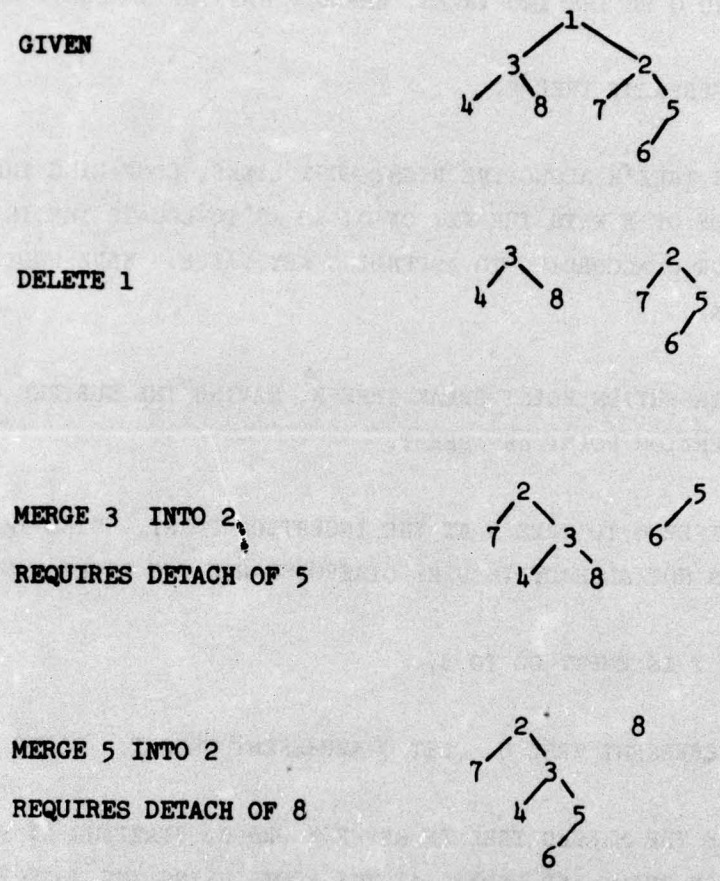
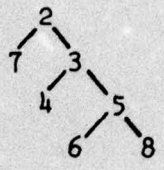
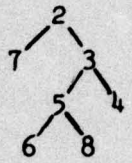


Figure 13. A Leftist Tree Example

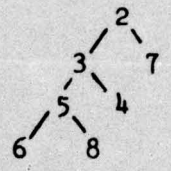
MERGE 8 INTO 2



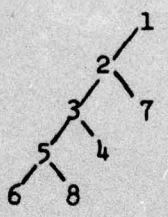
INTERCHANGE 4 AND 5



INTERCHANGE 7 AND 3



ADD 1



ORIGINAL TREE

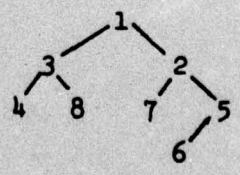


Figure 13. A Leftist Tree Example (Concluded)

DISTRIBUTION LIST

DEPARTMENT OF DEFENSE

Director
Armed Forces Radiobiology Research Institute
ATTN: Robert E. Carter
ATTN: Tech. Lib.

Director
Defense Communications Agency
ATTN: Code 930, Monte I. Burgett, Jr.
ATTN: Code 540.5

Defense Documentation Center
2 cy ATTN: TC

Director
Defense Nuclear Agency
ATTN: DDST
2 cy ATTN: STTL, Tech. Lib.

Headquarters
European Command
ATTN: ECJ6-PF

Commander
Field Command
Defense Nuclear Agency
ATTN: FCPR

Chief
Livermore Division, Field Command, DNA
ATTN: FCPRL

OJCS/J-3
ATTN: J-3, RDTA Br., WWMCCS, Plans Div.

DEPARTMENT OF THE ARMY

Commander
Frankford Arsenal
ATTN: SARFA-FCD, Marvin Elnick

Commander
Harry Diamond Laboratories
ATTN: AMXDO-EM, Raphael Wong
ATTN: AMXDO-FM, Robert F. Gray
ATTN: AMXDO-RBI, John A. Rosado
ATTN: AMXDO-RB, Joseph R. Miletta
ATTN: AMXDO-EM, R. Bostak
ATTN: AMXDO-RC, Robert B. Oswald, Jr.
ATTN: AMXDO-RCC, John E. Thompkins
ATTN: AMXDO-EM, J. W. Beilfuss

Commander
Picatinny Arsenal
ATTN: SARPA-ND-C-E, Amina Nordio
ATTN: SMUPA-FR-S-P, Lester W. Doremus
ATTN: SARPA-ND-N
ATTN: SARPA-AD-F-D, H. T. Rand

Commander
TRASANA
ATTN: ATAA-EAC, Francis N. Winans

DEPARTMENT OF THE ARMY (Continued)

Director
U.S. Army Ballistic Research Laboratories
ATTN: AMXBR-X, Julius J. Meszaros
ATTN: AMXRD-BVL, David L. Rigotti
ATTN: AMXBR-VL, John W. Kinch

Commander
U.S. Army Electronics Command
ATTN: AMSEL-GG-TD, W. R. Werk
ATTN: AMSEL-TL-IR, Edwin T. Hunter
ATTN: AMSEL-TL-MD, Gerhart K. Gaule

Commanding Officer
U.S. Army Electronics Command
ATTN: CPT Allan S. Parker

Commander
U.S. Army Electronics Proving Ground
ATTN: STEEP-MT-M, Gerald W. Durbin

Commandant
U.S. Army Field Artillery School
ATTN: ATSFA-CTD-ME, Harley Moberg

Commander
U.S. Army Mat. & Mechanics Research Center
ATTN: AMXMR-HH, John F. Dignam

Commander
U.S. Army Materiel Dev. & Readiness Command
ATTN: AMCRD-WN-RE, John F. Corrigan

Commander
U.S. Army Missile Command
ATTN: AMSI-RGP, Victor W. Ruwe
ATTN: AMSMI-RGP, Hugh Green
ATTN: AMSMI-RRR, Faison P. Gibson

Commander
U.S. Army Nuclear Agency
ATTN: ATCN-W, LTC Leonard A. Sluga

Project Manager
U.S. Army Tactical Data Systems, AMC
ATTN: Dwaine B. Huewe

Commander
White Sands Missile Range
ATTN: STEWS-TE-NT, Marvin P. Squires

DEPARTMENT OF THE NAVY

Chief of Naval Research
ATTN: Code 427

Commanding Officer
Naval Ammunition Depot
ATTN: Code 7024, James Ramsey

Commander
Naval Electronics Laboratory Center
ATTN: H. F. Wong

DEPARTMENT OF THE NAVY (Continued)

Commander
Naval Electronic Systems Command
ATTN: PME 117-21
ATTN: Code 50451
ATTN: ELEX 05323, Cleveland F. Watkins

Director
Naval Research Laboratory
ATTN: Code 2627, Doris R. Folen

Commander
Naval Surface Weapons Center
ATTN: Code 431, Edwin B. Dean
ATTN: Code WX-21, Tech. Lib.

Commander
Naval Surface Weapons Center
ATTN: Code FUR, Robert A. Amadori

Commanding Officer
Naval Weapons Evaluation Facility
ATTN: Code ATG, Mr. Stanley

Director
Strategic Systems Project Office
ATTN: SP-2701, John W. Pitsenberger

DEPARTMENT OF THE AIR FORCE

Commander
Aeronautical Systems Division, AFSC
ATTN: ASD-YH-EX, Lt Col Robert Leverette
ATTN: ENESS, Mr. Marth

AF Aero-Propulsion Laboratory, AFSC
ATTN: POD, P. E. Stover

AF Weapons Laboratory, AFSC
ATTN: ELP, Carl E. Baum
ATTN: ELA
ATTN: ELC
4 cy ATTN: SUL
ATTN: HO
15 cy ATTN: ELP, R. Simon

AFTAC
ATTN: TAE

Air Force Avionics Laboratory, AFSC
ATTN: AFAL, TEA, Hans J. Hennecke
ATTN: AFAL, AAA

Commander
Foreign Technology Division, AFSC
ATTN: FTD/PDJC

AF Contract Management Division
ATTN: HO, Historian

Commander
Ogden Air Logistics Center
ATTN: MMEWM, Robert Joffa

SAMSO/DY
ATTN: DYS, Maj Larry A. Darda

SAMSO/YD
ATTN: YDD, Maj Marion F. Schneider

DEPARTMENT OF THE AIR FORCE (Continued)

Commander in Chief
Strategic Air Command
ATTN: XPFS, Maj Brian G. Stephan

ENERGY RESEARCH & DEVELOPMENT ADMINISTRATION

University of California
Lawrence Livermore Laboratory
ATTN: Lawrence Cleland, L-156
ATTN: E. K. Miller, L-156
ATTN: Donald J. Meeker, L-153
ATTN: William J. Hogan, L-531
ATTN: Frederick R. Kovar, L-94

Los Alamos Scientific Laboratory
ATTN: Doc. Con. for Bruce W. Noel
ATTN: Doc. Con. for J. Arthur Freed

Sandia Laboratories
ATTN: D. S. Hill, Org. 2116
ATTN: R. Gregory, Org. 2140
ATTN: C. Gwyn, Org. 2142

DEPARTMENT OF DEFENSE CONTRACTORS

Aerojet Electro-Systems Co. Div.
ATTN: Thomas D. Hanscome

Aeronutronic Ford Corporation
ATTN: Ken C. Attinger
ATTN: E. R. Poncelet, Jr.
ATTN: Tech. Info. Section

Aeronutronic Ford Corporation
ATTN: Samuel R. Crawford, M.S. 531

Avco Research & Systems Group
ATTN: Research Library, A-830, Rm. 7201

The BDM Corporation
ATTN: Allan F. Malmberg
ATTN: Larry D. Ray
ATTN: T. H. Neighbors
ATTN: William Druen
5 cy ATTN: D. Alexander

The Bendix Corporation
ATTN: Doc. Con.

The Bendix Corporation
Research Laboratories Division
ATTN: Donald J. Niehaus, Mgr. Prgm. Dev.

The Boeing Company
ATTN: Robert S. Caldwell, 2R-00
ATTN: Howard W. Wicklein, 17-11
ATTN: Aerospace Library
ATTN: David L. Dye, 87-75
ATTN: Donald W. Egelkrout, 2R-00

Booz-Allen & Hamilton, Inc.
ATTN: Raymond J. Chrisner

Brown Engineering Company, Inc.
ATTN: David L. Lambert, M.S. 18

Charles Stark Draper Laboratory, Inc.
ATTN: Paul R. Kelly
ATTN: Kenneth Fertig

DEPARTMENT OF DEFENSE CONTRACTORS (Continued)

Computer Sciences Corporation
ATTN: Richard H. Dickhaut

Cutler-Hammer, Inc.
AIL Division
ATTN: Anne Anthony, Central Tech. Files

University of Denver
Colorado Seminary
ATTN: Sec. Officer for Fred P. Venditti

The Dikewood Corporation
ATTN: L. Wayne Davis

E-Systems, Inc.
Greenville Division
ATTN: Library, 8-50100

Exp. & Math. Physics Consultants
ATTN: Thomas M. Jordan

Fairchild Industries, Inc.
ATTN: Mgr., Config. Data & Standards

The Franklin Institute
ATTN: Ramie H. Thompson

Fairchild Semiconductor
ATTN: David K. Myers

Garrett Corporation
ATTN: Robert E. Weir, Dept. 93-9

General Electric Company
Space Division
ATTN: John R. Greenbaum
ATTN: Larry I. Chasen
ATTN: Joseph C. Peden, CCF 8301
ATTN: John L. Andrews
ATTN: James P. Spratt

General Electric Company
Re-Entry & Environmental Systems Div.
ATTN: Robert V. Benedict

General Electric Company
TEMPO-Center for Advanced Studies
ATTN: Royden R. Rutherford
ATTN: DASIAC
ATTN: William McNamera
ATTN: M. Espig

General Electric Company
ATTN: CSP 0-7, L. H. Dee

General Electric Company
Aerospace Electronics Systems
ATTN: W. J. Patterson, Drop 233
ATTN: George Francis, Drop 233

General Electric Company-TEMPO
ATTN: DASIAC for William Alfonte

General Research Corporation
ATTN: Robert D. Hill

General Research Corporation
Washington Operations
ATTN: David K. Osias

DEPARTMENT OF DEFENSE CONTRACTORS (Continued)

GTE Sylvania, Inc.
Electronics Systems Group-Eastern Division
ATTN: Leonard L. Blaisdell
ATTN: James A. Waldon

GTE Sylvania, Inc.
ATTN: Herbert A. Ullman
ATTN: David P. Flood
ATTN: Charles H. Ramsbottom

Gulton Industries, Inc.
Engineered Magnetics Division
ATTN: Eng. Magnetics Div.

Harris Corporation
Harris Semiconductor Division
ATTN: Carf F. Davis, M.S. 17-220
ATTN: Wayne E. Abare, M.S. 16-111
ATTN: T. L. Clark, M.S. 4040
ATTN: J. Cornell

Hazeltine Corporation
ATTN: Tech. Info. Ctr., M. Waite

Honeywell, Incorporated
Government & Aeronautical Products Division
ATTN: Ronald R. Johnson, A-1622

Honeywell, Incorporated
Aerospace Division
ATTN: Stacey H. Graff, M.S. 725-J
ATTN: James D. Allen, M.S. 775-D
ATTN: Harrison H. Noble, M.S. 725-5A

Honeywell, Incorporated
Radiation Center
ATTN: Tech. Lib.

Hughes Aircraft Company
ATTN: Kenneth R. Walker, M.S. D-157
ATTN: Billy W. Campbell, M.S. 6-E-110

Hughes Aircraft Company
Space Systems Division
ATTN: Edward C. Smith, M.S. A-620
ATTN: William W. Scott, M.S. A-1080

IBM Corporation
ATTN: Frank Frankovsky
ATTN: Harry W. Mathers, Dept. M-41

IIT Research Institute
ATTN: Irving N. Mindel

Intelcom Rad Tech
ATTN: MDC
ATTN: Eric P. Wenaas
ATTN: R. L. Mertz
ATTN: Leo D. Cotter
ATTN: Michael Chipman

Kaman Sciences Corporation
ATTN: Albert P. Bridges
ATTN: Walter E. Ware
ATTN: W. Foster Rich
ATTN: John R. Hoffman
ATTN: Donald H. Bryce

DEPARTMENT OF DEFENSE CONTRACTORS (Continued)

Litton Systems, Inc.
Guidance & Control Systems Division
ATTN: Val J. Ashby, M.S. 67
ATTN: R. W. Maughmer

Lockheed Missiles & Space Co., Inc.
ATTN: Benjamin T. Kimura, Dept. 81-14
ATTN: George F. Heath, Dept. 81-14
ATTN: Hans L. Schneemann, Dept. 81-64

LTV Aerospace Corporation
ATTN: Tech. Data Ctr.

Martin Marietta Aerospace
Orlando Division
ATTN: William W. Mras, MP-413
ATTN: Mona C. Griffith, Library, MP-30
ATTN: Jack M. Ashford, MP-537

Martin Marietta Corporation
Denver Division
ATTN: J. E. Goodwin, Mail 0452
ATTN: Paul G. Kase, Mail 8203

McDonnell Douglas Corporation
ATTN: Tom Ender
ATTN: Tech. Lib.

McDonnell Douglas Corporation
ATTN: Stanley Schneider
ATTN: Raymond J. DeBattista

McDonnell Douglas Corporation
ATTN: Tech. Lib., C1-290/36-84

Mission Research Corporation
ATTN: William C. Hart

Mission Research Corporation
ATTN: David E. Merewether

Mission Research Corporation-San Diego
ATTN: V. A. J. Van Lint

The Mitre Corporation
ATTN: M. E. Fitzgerald

National Academy of Sciences
ATTN: National Materials Advisory Board for
R. S. Shane, Nat. Materials Advsy.

Northrop Corporation
Electronic Division
ATTN: Boyce T. Ahlport
ATTN: Vincent R. DeMartino
ATTN: George H. Towner

Northrop Corporation
ATTN: Joseph Srour
ATTN: David N. Pocock
ATTN: Orlie L. Curtis, Jr.

Northrop Corporation
Electronic Division
ATTN: Joseph D. Russo

Physics International Company
ATTN: Doc. Con. for John H. Huntington

DEPARTMENT OF DEFENSE CONTRACTORS (Continued)

R & D Associates
ATTN: Leonard Schlessinger
ATTN: S. Clay Rogers

The Rand Corporation
ATTN: Cullen Crain

Raytheon Company
ATTN: Gajanan H. Joshi, Radar Sys. Lab.

Raytheon Company
ATTN: Harold L. Flescher
ATTN: James R. Weckback

RCA Corporation
Government & Commercial Systems
ATTN: George J. Brucker

RCA Corporation
ATTN: E. Van Keuren, 13-5-2

Rockwell International Corporation
ATTN: George C. Messenger, FB-61
ATTN: James E. Bell, HA-10

Rockwell International Corporation
Electronics Operations
ATTN: Dennis Sutherland
ATTN: Alan A. Langenfeld
ATTN: Mildred A. Blair

Sanders Associates, Inc.
ATTN: Moe L. Aitel, NCA 1-3236

Science Applications, Inc.
ATTN: Frederick M. Tesche

Science Applications, Inc.
ATTN: William L. Chadsey

Science Applications, Inc.
ATTN: J. Robert Beyster
ATTN: Larry Scott

Science Applications, Inc.
Huntsville Division
ATTN: Noel R. Byrn

Science Applications, Inc.
ATTN: J. Roger Hill

Simulation Physics, Inc.
ATTN: John R. Ugtum

The Singer Company
ATTN: Irwin Goldman, Eng. Management

Sperry Flight Systems Division
Sperry Rand Corporation
ATTN: D. Andrew Schow

Sperry Rand Corporation
Sperry Division
ATTN: Paul Marraffino

Stanford Research Institute
ATTN: Arthur Lee Whitson
ATTN: Philip J. Dolan

DEPARTMENT OF DEFENSE CONTRACTORS (Continued)

Sundstrand Corporation
ATTN: Curtis B. White

Systron-Donner Corporation
ATTN: Gordon B. Dean

Texas Instruments, Inc.
ATTN: Donald J. Manus, M.S. 72
ATTN: Gary F. Hanson
ATTN: Pradeep Shah, M.S. 82
ATTN: Stephen A. Evans, M.S. 82

Texas Tech University
ATTN: Travis L. Simpson

TRW Systems Group
ATTN: Aaron H. Narevsky, R1-2144
ATTN: A. A. Witteles, R1-1120
ATTN: Benjamin Sussholtz
ATTN: Lillian D. Singletary, R1-1070
ATTN: A. M. Liebschutz, R1-1162
ATTN: Richard H. Kingsland, R1-2154
ATTN: Jerry I. Lubell

DEPARTMENT OF DEFENSE CONTRACTORS (Continued)

TRW Systems Group
San Bernardino Operations
ATTN: John E. Dahnke
ATTN: H. S. Jensen

United Technologies Corporation
Hamilton Standard Division
ATTN: Raymond G. Giguere

Westinghouse Electric Corporation
ATTN: Henry P. Kalapaca, M.S. 3525

Westinghouse Defense & Electronic Systems Center
Advanced Technology Labs.
ATTN: J. Ronald Cricchi

University of Florida
ATTN: F. A. Lindholm

University of South Florida
Department of Electrical Engineering
ATTN: J. Bowers

AD-A031 802 BDM CORP ALBUQUERQUE N MEX F/G 9/2
SCEPTRE/LOGIC CIRCUIT ANALYSIS PROGRAM USER'S MANUAL.(U)
JUL 76 A F MALMBERG, L D RAY F29601-74-C-0106
UNCLASSIFIED BDM/A-145-75-TR-R1 AFWL-TR-75-279 NL

2 OF 2
AD
A031 802

END

DATE
FILMED
7-77

SUPPLEMENTARY

INFORMATION

AD-A031 802

AIR FORCE WEAPONS LABORATORY
Air Force Systems Command
Kirtland Air Force Base
New Mexico 87117

2 February 1977

ERRATA

AFWL-TR-75-279 SCEPTRE/LOGIC CIRCUIT ANALYSIS PROGRAM USER'S MANUAL, July 1976 (UNCLASSIFIED).

Add Reference AFWL-TR-71-183, Modeling of Integrated Circuits as Components of Large Systems, by D. N. Pocock, et al., Northrop Corporation, Dec 1972.

Authority:
ROBERT G. SIMON
Project Officer
Phenomenology & Technology Branch

E. L. Elliott
E. L. ELLIOTT
Chief, Reports & Graphics Branch